ETSI Informáticos, UPM

Control 1 Grupos 3S1M-B y 3S5T

Tiempo: 2 horas

Apellidos ..... **Nombre** .....

1) (2 puntos)

a) Estudia si la sucesión 5,3,3,2,2,1 puede ser gráfica. En caso afirmativo construye un grafo cuya sucesión sea la dada mediante el algoritmo descrito en clase.

El algoritmo de Havel-Hakimi responde si la sucesión es o no gráfica a la vez que, si lo es, proporciona un grafo cuya sucesión de grados es la dada:

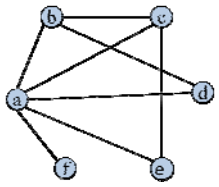
$5_a\ 3_b\ 3_c\ 2_d\ 2_e\ 1_f$

$2_b\ 2_c\ 1_d\ 1_e\ 0_f$

$1_c\ 0_d\ 1_e\ 0_f$

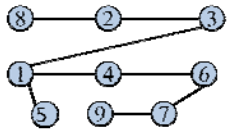
$1_c\ 1_e\ 0_d\ 0_f$

$0_e\ 0_d\ 0_f$



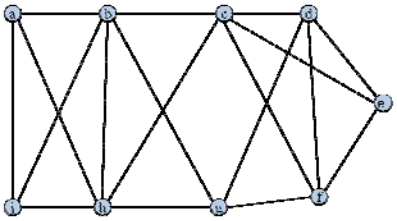
b) Construye el árbol etiquetado cuyo código de Prüfer es (1, 2, 3, 1, 4, 6, 7). Encuentra su centro mediante el algoritmo descrito en clase.

El árbol pedido es el de la figura y su centro se encuentra eliminando progresivamente las hojas hasta que quedan uno o dos vértices. En este caso es el árbol inducido por los vértices $\{1,4\}$.



2) (2,5 puntos)

Sea G el grafo de la figura



a) Estudia si el grafo es bipartido.

No lo es porque contiene 3-ciclos como, por ejemplo, $\mathrm{a}-\mathrm{b}-\mathrm{h}-\mathrm{a}$.

b) Estudia su conectividad por vértices y por aristas.

La conectividad por vértices es 2 porque al eliminar b y h el grafo resultante no es conexo pero al eliminar un vértice cualquiera el grafo sigue siendo conexo

La conectividad por aristas es 3 porque al eliminar dos aristas cualesquiera el grafo sigue siendo conexo y al eliminar, por ejemplo las tres aristas incidentes con a deja de serlo

c) Aplica el algoritmo de búsqueda en profundidad para construir un árbol generador de G.

Comenzando en a, $V_r = \{a, b, c, d, e, f, g, h, j\}$ $V_n = \{\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, \cancel{e}, \cancel{f}, \cancel{g}, \cancel{h}, \cancel{j}\}$

$P = [\cancel{a}, \cancel{b}, \cancel{c}, \cancel{d}, \cancel{e}, \cancel{f}, \cancel{g}, \cancel{h}, \cancel{j}]$

$Aristas = \{ab, bc, cd, de, ef, fg, gh, hj\}$

d) Halla una orientación de G tal que el digrafo obtenido sea fuertemente conexo.

El algoritmo utilizado en clase consiste en aplicar una búsqueda en profundidad donde las aristas que forman el árbol de búsqueda se orientan de la raíz (vértice en el que comienza la búsqueda) hacia los nuevos vértices visitados, mientras que el resto de las aristas que no están en el árbol de búsqueda se orientan hacia la raíz:

3) (2,5 puntos)

Sea D el digrafo de vértices {*a*, *b*, *c*, *d*, *e*, *f*, *g*} cuya matriz de pesos es M. Se quiere calcular el camino más corto del vértice *e* al vértice *h*.

$$\begin{pmatrix} 0 & 2 & \infty & 5 & \infty & 1 & \infty & \infty \\ \infty & 0 & 3 & \infty & 2 & \infty & 1 & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty & 4 \\ \infty & 1 & \infty & 0 & 4 & \infty & \infty & \infty \\ \infty & \infty & 2 & \infty & 0 & 3 & 2 & 7 \\ \infty & 5 & \infty & \infty & \infty & 0 & 1 & 1 \\ \infty & 6 & \infty & \infty & \infty & \infty & 0 & 4 \\ \infty & \infty & 3 & \infty & \infty & 4 & \infty & 0 \end{pmatrix}$$

a) Razona cuál es el (o uno de los) algoritmo(s) más adecuado(s) para resolverlo.

Como todos los pesos son positivos, el algoritmo más adecuado para resolver este problema es el algoritmo de Dijkstra, ya que es el más eficiente (el de menor complejidad)

b) Ejecuta el algoritmo y describe el camino y la distancia de *e* a *h*.

Nota: No se permite el uso de dispositivos electrónicos.

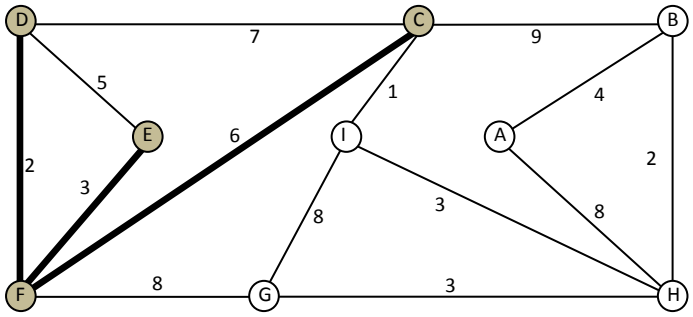
D*a*	D*b*	D*c*	D*d*	D*e*	D*f*	D*g*	D*h*		p*a*	p*b*	p*c*	p*d*	p*e*	p*f*	p*g*	p*h*
∞	∞	∞	∞	**0**	∞				-	-	-	-	**-1**	-		
∞	∞	**2**	∞		3	2	7		-	-	**e**	-		e	e	e
∞	∞		∞		3	**2**	6		-	-		-		e	**e**	c
∞	8		∞		**3**		6		-	g		-		**e**		c
∞	8		∞				**4**		-	g		-				**f**

Se fija el vértice objetivo *h*, por lo que el camino de *e* a *h* no puede mejorar. Fin del algoritmo.
El camino es *e-f-h* y la distancia 4

c) Explica con detalle la última iteración del algoritmo que produce modificaciones en los datos y el método para obtener el camino.
El vértice a menor distancia de *e* entre los no fijados es *f*, por lo que se fija y se toma como vértice activo. Sus adyacentes entre los no fijados son *b* y *h*:
¿ $8 = d(b) > d(f) + w(\overrightarrow{fb}) = 3 + 5$? La respuesta es negativa por lo que no hay modificaciones.
¿ $6 = d(h) > d(f) + w(\overrightarrow{fh}) = 3 + 1$? La respuesta es afirmativa por lo que $d(h) \leftarrow 4$ y $p(h) \leftarrow f$.
Para recuperar el camino, se comienza en el vértice objetivo: *h*. En la tabla de predecesores se ve que el predecesor de *h* es *f*. El predecesor de *f* es *e* y el de *e* *-1* que indica que es el vértice origen. Por tanto, el camino será *e-f-h*.

4) (3 puntos)

a) En un grafo cuyas aristas tienen pesos positivos, ¿coinciden el árbol generador de peso mínimo y el árbol de caminos mínimos desde un vértice dado a todos los demás? En caso afirmativo razona por qué y en caso negativo muestra un ejemplo donde no coincidan. Di qué algoritmos se utilizan para resolver estos problemas.



En general no coinciden el árbol de peso mínimo y el de caminos mínimos. Por ejemplo, en el grafo de la figura, la arista FG no está en el árbol de peso mínimo, pero sí es el camino más corto del vértice F al G, por lo que sí estará en el árbol de caminos mínimos de F a los demás vértices del grafo.
Para construir el árbol de peso mínimo se utilizan los algoritmos de Prim y de Kruskal. Para el de caminos mínimos el de Dijkstra ya que es el de menor complejidad para pesos positivos.

b) Especifica el control de parada para cada uno de los algoritmos del apartado anterior.
El algoritmo de Kruskal para cuando el contador que almacena el número de aristas añadidas llega a n-1 (inicializándolo en 0) o a n (inicializándolo en 1).
El algoritmo de Prim para cuando se han añadido al árbol todos los vértices ($V_n = \emptyset$) o, lo que es lo mismo, el contador que almacena el número de vértices añadidas llega a n-1 (inicializándolo en 0) o a n (inicializándolo en 1).
El algoritmo de Dijkstra para cuando se fija el vértice objetivo o, si no lo hay, cuando no quedan vértices sin fijar.

c) En el grafo de la figura, se han aplicado varias iteraciones de un algoritmo que construye un árbol generador de peso mínimo. Razona de cuál se trata y describe con detalle el estado actual de las estructuras de datos del algoritmo y dos iteraciones más.

d) Se ha aplicado el algoritmo de Prim, si hubiera sido el de Kruskal, la primera arista que se habría añadido habría sido CI que tiene peso 1
Estado actual de las estructuras: Aristas: {DF, FE, FC} y primera fila de la tabla de resolución.

V_r C, D, E, F, I, H					
V_n	A	B	G	H	I
	∞	9 C	8 F	∞	**1** **C**
	∞	9 C	8 F	**3** **I**	
	8 H	2 H	3 H		

Primera iteración: Se elige el vértice con menor valor: I, se elimina de V_n y se añade a V_r. Se añade CI a Aristas. Se actualizan los valores para los adyacentes a I en V_n: G y H,

- ¿8 > w(IG)? No, por tanto no se cambia
- ¿∞ > w(IH)? Sí, por tanto los valores de H se actualizan a [3,I]

Segunda iteración: Se Se elige el vértice con menor valor: H, se elimina de V_n y se añade a V_r. Se añade IH a Aristas. Se actualizan los valores para los adyacentes a H en V_n: A, B y G,

- ¿∞ > w(HA)?Sí, por tanto los valores de A se actualizan a [8,H]
- ¿9 > w(HB)? Sí, por tanto los valores de B se actualizan a [2,H]
- ¿8 > w(HG)? Sí, por tanto los valores de G se actualizan a [3,H]